Санкт-Петербургский государственный университет

Кафедра информатики

Березин Алексей Иванович

АЛГЕБРАИЧЕСКИЕ БАЙЕСОВСКИЕ СЕТИ: СИСТЕМА АНАЛИЗА И СИНТЕЗА ВТОРИЧНОЙ СТРУКТУРЫ

Бакалаврская работа

Научный руководитель:
профессор кафедры информатики, д. ф.-м. н., доцент Тулупьев А. Л.

Рецензент:
доцент департамента прикладной математики
и бизнес-информатики НИУ ВШЭ СПб, к.ф.-м.н. Сироткин А. В.

Санкт-Петербург
2016

SAINT-PETERSBURG STATE UNIVERSITY

Computer Science department

Alexey Berezin

ALGEBRAIC BAYESIAN NETWORKS: ANALYSIS AND SYNTHESIS SYSTEM OF SECONDARY STRUCTURES

Bachelor's Thesis

Scientific supervisor:
Professor, Computer Science Department, PhD, Associate Professor Tulupyev A. L.

Reviewer:
Associate Professor, NRU HSE, PhD Sirotkin A. V.

Saint-Petersburg
2016

Оглавление

Введение

Актуальность темы исследования. В 1993 году В.И. Городецкий ввел понятие теории алгебраических байесовских сетей (АБС), которая является экземпляром класса вероятностных графических моделей (ВГМ) [9, 22, 23, 30–32].

АБС представлена в виде ненаправленного графа, вершины которого являются фрагментами знаний [8]. Существуют две основные задачи в рамках АБС: проведение логико-вероятностного вывода [7, 10, 21, 24, 25, 28, 29] и генерация глобальных структур.

Структурный анализ изучает алгоритмы генерации глобальных структур: первичной, вторичной, третичной и четвертичной [10]. Результаты предыдущих исследований, посвященных первичной структуре, предложены в [14, 17, 18]. Синтез вторичной структуры осуществляется и в настоящее время. Проделанную работу по вторичной структуре можно изучить в [6, 20]. Генерация множества вторичной структуры, или множества минимальных графов смежности (ММГС), образует класс независимых алгоритмов [16, 19, 28]. Выделение единственной структуры из такого множества может производиться на основе численных характеристик логико-вероятностного вывода. Также исследования в данный момент ведутся и над третичной [13] и четвертичной структурами [17, 20]. Представление по данным структурам можно получить в [27].

Объектом исследования выступает алгебраическая байесовская сеть, а **предметом исследования** — генерация множества минимальных графов смежности в данной теории.

Целью данной выпускной квалификационной работы является ускорение автоматизации синтеза множества вторичной структуры. Для достижения данной цели необходимо решить **следующие задачи**:

1. Перенос программных достижений на язык C#, касающихся алгоритмов синтеза ММГС.

2. Разработка инкрементальных алгоритмов генерации ММГС, доказательство их корректности и эффективности, а также предоставление реализации алгоритмов на языке C#.

3. Осуществление визуализации полученного множества.

Методы исследования. Для решения поставленных задач в данной области понадобилось изучить методические материалы, поставить проблему, проанализировать ее, спроектировать несколько возможных вариантов решения. Затем были выбраны подходящие средства и технологии программирования, связанные с языком реализации (C#), средой разработки (Visual Studio), сервисом для совместной разработки (BitBucket). После чего стало необходимым провести тестовые случаи и статистические эксперименты с целью обоснования корректности и эффективности полученного решения. В качестве методов решения используются комбинаторика, теория графов, методы объектно-ориентированного программирования (ООП), эмпирические (постановка и проведение статистических экспериментов) и теоретические методы (анализ предыдущих результатов, синтез и формализация разработанных алгоритмов).

Научная новизна. В данной выпускной квалификационной бакалаврской работе формализованы и исследованы новые инкрементальные алгоритмы синтеза множества минимальных графов смежности, что отличает указанный способ генерации от предыдущих вариантов и позволяет с его помощью быстро создавать множество вторичной структуры в ситуации постоянного изменения первичной структуры.

Апробация результатов. Результаты исследования докладывались на следующих научных мероприятий:

1. Всероссийская научная конференция по проблемам информатики СПИСОК, Санкт-Петербург, апрель 2016.

2. Международная конференция Intelligent Information Technologies for Industry, Сочи, май 2016.

Теоретическая и практическая значимость исследования. Результаты исследования развивают теоретическую и технологическую базу, в частности, разработаны и реализованы новые методы генерации ММГС, что вносит теоретическую значимость и технологический вклад в раздел теории АБС.

Публикации. По теме данной работы было сделано 2 публикации, одна из которых была принята к публикации в Scopus.

Личный вклад А. И. Березина в публикациях с соавторами характеризуется следующим образом: в [28] — краткое описание существующих алгоритмов и визуализация промежуточного множества, в [2] — декрементальный алгоритм генерации множества минимальных графов смежности, в [3] — представление проблем генерации ММГС и методы их устранения, в [1] — описание архитектуры комплекса программ и приведение примера получающегося множества.

Структура и объем работы. Работа состоит из введения, пяти глав, заключения, используемой литературы и иллюстраций.

Первая глава носит обзорный характер и состоит из четырех разделов. В первом разделе описаны основные понятия и определения, введенные в предыдущих работах. Во втором разделе подробно рассмотрена классификация сепараторов, которая будет использоваться во всей работе. В третьем разделе изложена схема алгоритмов синтеза подмножеств множества минимальных графов смежности. Четвертая глава приводит обозначения, которые применяются в псевдокодах алгоритмов.

Вторая глава охватывает описание двухфазных алгоритмов, которые были разработаны в предыдущих работах. В первом разделе представляются MK-алгоритмы. Во втором разделе приводится алгоритм сборки.

Третья глава содержит теоретические результаты, полученные соискателем. В первом разделе рассматривается инкрементальный MK-алгоритм. Во втором раздел описывает декрементальный MK-алгоритм.

Четвертая глава содержит описание сравнительного анализа разработанных и существующих алгоритмов. Первый раздел представляет

схема проведения экспериментов. Второй раздел приводит результаты этих экспериментов.

Пятая глава содержит описание системы синтеза и анализа, части программного комплекса, реализующего алгоритмы из второй и третьей глав. Первый раздел представляет архитектуру данной системы. Во втором разделе предложены примеры работы системы.

Общий объем работы составляет 46 страниц. Она содержит 14 рисунков и 12 листингов алгоритмов. Список используемой литературы содержит 33 источника.

Эта работа является частью более широких инициативных проектов, выполняющихся в лаборатории теоретических и междисциплинарных основ информатики СПИИРАН под руководством А.Л. Тулупьева; кроме того, разработки были частично поддержаны грантами РФФИ 15-01-09001-а — «Комбинированный логико-вероятностный графический подход к представлению и обработке систем знаний с неопределенностью: алгебраические байесовские сети и родственные модели», 12-01-00945-а —«Развитие теории алгебраических байесовских сетей и родственных им логико-вероятностных графических моделей систем знаний с неопределенностью», 09-01-00861-а —«Методология построения интеллектуальных систем поддержки принятия решений на основе баз фрагментов знаний с вероятностной неопределенностью», а также проектом по ФЦНТП «Исследования и разработки по приоритетным направлениям развития науки и техники на 2002–2006 годы», 2006-РИ-19.0/001/211, Государственный контракт от «28» февраля 2006 г., No 02.442.11.7289, «Направленные циклы в байесовских сетях доверия: вероятностная семантика и алгоритмы логико-вероятностного вывода для программных комплексов с байесовской интеллектуальной компонентой».

1. Основные объекты и результаты теории алгебраических байесовских сетей

Введение

В этой главе будут представлены базовые объекты теории алгебраический байесовских сетей (АБС), которые уже сформировались в предыдущих работах [15, 16, 19], новые определения, введенные во время разработки алгоритмов, включая [27], и обозначения, используемые в псевдокодах реализованных алгоритмов.

1.1. Элементы глобальных структур

***Определение* 1.1.** [15, 19] *Нагруженным графом называется тройка* $\langle G, A, W \rangle$*, где* G *— ненаправленный граф,* A *— алфавит,* W *— функция нагрузки, заданная на вершинах и ребрах графа* G*, принимающая значения из* 2^A*.*

***Определение* 1.2.** [15, 16, 19] *Сепаратором двух вершин в нагруженном графе называется пересечение нагрузок соответствующих вершин:*

$$\mathrm{Sep}_{u,v} = W_u \cap W_v.$$

***Определение* 1.3.** [15, 19] *Нагруженный граф согласован тогда и только тогда, когда*

$$\forall e = \{u, v\}\ W_e = \mathrm{Sep}_{u,v}.$$

***Определение* 1.4.** [19] *Под первичной структурой будет пониматься набор взаимно непоглощающихся подалфавитов* A*:*

$$\forall w_i, w_j \in W, W \subset A, i \neq j \Rightarrow w_i \not\subseteq w_j.$$

Семейство первичных структур обозначим как PrimaryStructures.

***Определение* 1.5.** [15, 19] *Две вершины называются сочлененными, если их сепаратор непуст.*

***Определение* 1.6.** [15, 19] *Граф максимальных фрагментов знаний (граф МФЗ) — граф, который является согласованным и в котором ребра возможны только между сочлененными вершинами.*

***Определение* 1.7.** [15,16,19] *Магистральный путь между двумя вершинами u и v в согласованном нагруженном графе — путь вида*

$$u = e_0, e_1, \ldots, e_{n-1}, e_n = v : \forall e_i : Sep_{u,v} \subseteq W_{e_i}.$$

***Определение* 1.8.** [15, 16, 19] *Графом смежности называется магистрально связный граф МФЗ.*

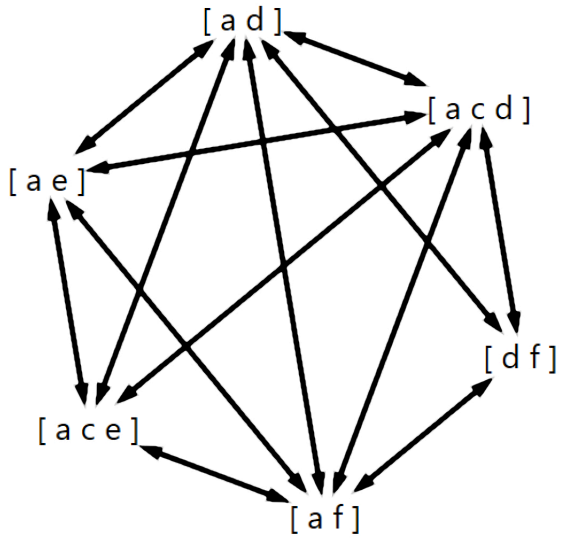


Рис. 1: Пример графа смежности.

***Определение* 1.9.** *Сужением вершин и ребер исходного графа $G = \langle V, E\rangle$ на сепаратор S будем называть такие вершины и ребра исходного графа, нагрузки которых содержат или равны S:*

$$V_{S\subseteq} = \{V_i | V_i \in V, S \subseteq W(V_i)\}, E_{S\subseteq} = \{E_i | E_i \in E, S \subseteq W(E_i)\}.$$

***Определение* 1.10.** *Сильным сужением ребер исходного графа $G = \langle V, E\rangle$ на сепаратор S будем называть сужение ребер на сепаратор S,*

из которого были удалены все ребра с нагрузкой S:

$$E_{S\subset} = \{E_i | E_i \in E, S \subset W(E_i)\}.$$

***Определение* 1.11.** [15] *Сужение $G \downarrow S$ согласованного лексического графа G на сепаратор S — ненаправленный граф, полученный сужением его вершин и ребер на сепаратор S:*

$$G \downarrow S = \langle V_{S\subseteq}, E_{S\subseteq} \rangle.$$

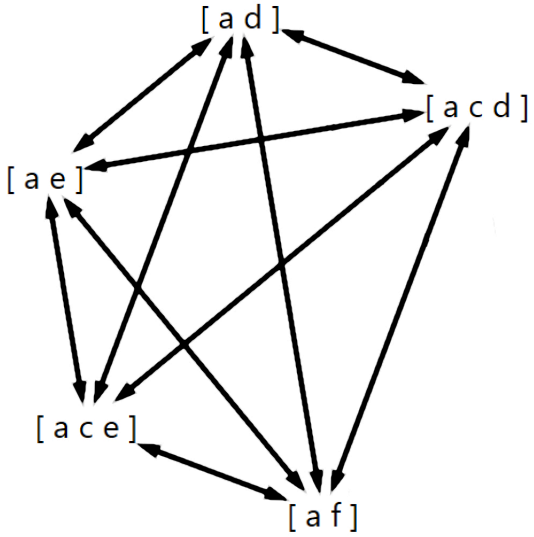


Рис. 2: Пример сужения по графу смежности, представленному на Рис. 1, на сепаратор $s = [\ a\ ]$.

***Определение* 1.12.** [15] *Сильное сужение $G \downarrow\!\downarrow S$ согласованного лексического графа G на сепаратор S — ненаправленный граф, вершины которого получены сужением, а ребра — сильным сужением на сепаратор S:*

$$G \downarrow\!\downarrow S = \langle V_{S\subseteq}, E_{S\subset} \rangle.$$

Если явно не указано, по какому графу построено соответствующее сужение, будем понимать сужение максимального графа смежности, графа смежности, число ребер которого максимально.

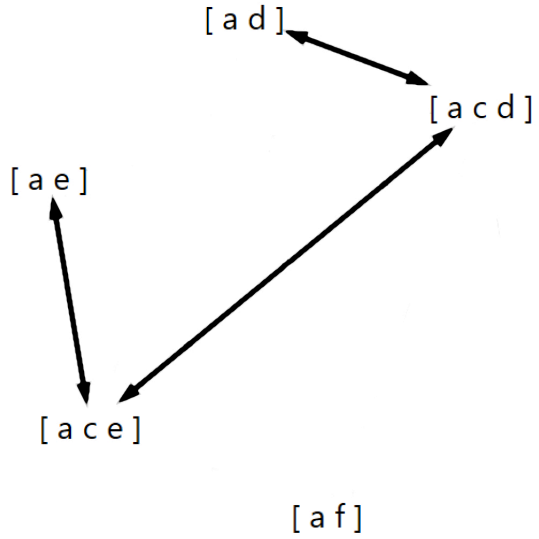
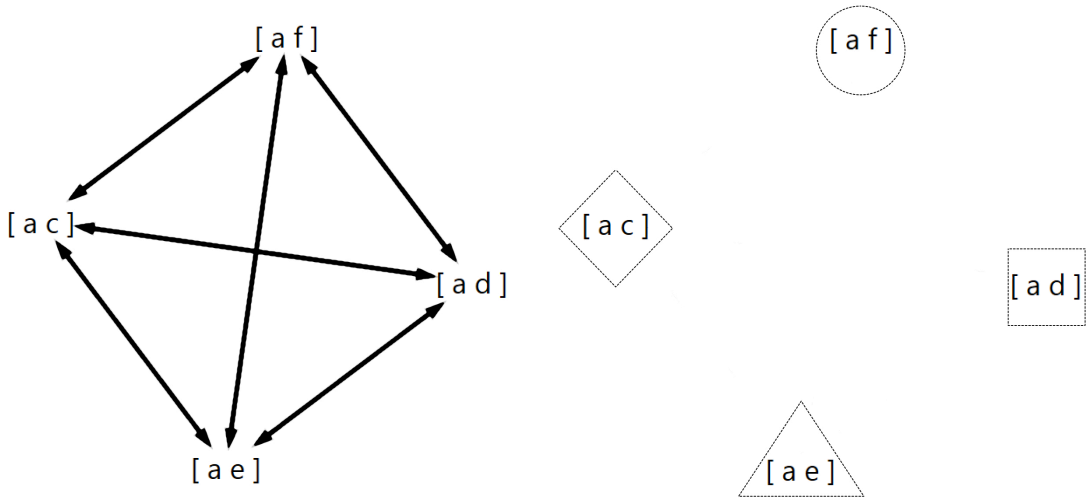


Рис. 3: Пример сильного сужения по графу смежности на Рис. 1 на сепаратор $s = [\ a\ ]$.

***Определение* 1.13.** [15] *Владениями сепаратора S по графу G будут называться компоненты связности сильного сужения по заданному графу на сепаратор S.*



(a) Первичная структура. (b) Четыре владения.

Рис. 4: Пример владений для сепаратора $S = [\ a\ ]$.

***Определение* 1.14.** [15] *Феод сепаратора S — f_i^S — вершина, получившаяся сжатием какого-то владения сепаратора S.*

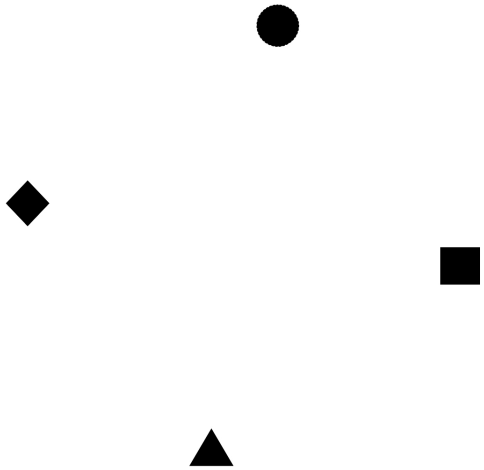


Рис. 5: Пример феодов по владениям из Рис. 4b.

***Определение* 1.15.** [15] *Курия нагрузки S — K_S — мультиграф, полученный сужением $G \downarrow S$ графа G на сепаратор S.*

***Определение* 1.16.** [15] *Оммаж H_S — курия K_S, являющаяся деревом, все ребра которой имеют кратность, равную единице.*

***Определение* 1.17.** [15] *Жила $s^S_{i,j}$ — граф, построенный на вершинах суженния $G \downarrow S$ и соответствующий оммажу H^S_i : $\sigma_U(s^S_{i,j}) = H^S_i$. Множество жил сепаратора S будем обозначать как* $\mathrm{SinewSet}_S$

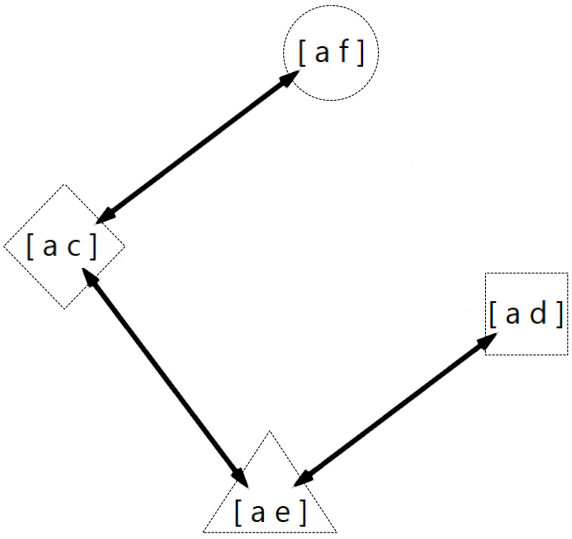


Рис. 6: Пример жилы над владениями из Рис. 4b по сепаратору $S = [\, a \,]$.

***Определение* 1.18.** [15] *Пучок — граф, построенный путем объединения жил, выбранных по одной для каждого сепаратора:*

$$\bigcup_{S_k \in Sep} s^{S_k}_{i_k,j_k}, \text{ где } s^{S_k}_{i_k,j_k} \in \text{SinewSet}_{S_k}.$$

Если рассматривать множество пучков для графа, представленного на Рис. 4а, то оно будет совпадать с множество жил, т.к. во вторичной структуре присутствует только один сепаратор $S = [\ a\ ]$.

1.2. Классификация сепараторов

Рассмотрим упрощенную классификацию сепараторов, введенную в работе [19], которая понадобится в следующих разделах.

***Определение* 1.19.** [19] *Моносепаратор* $S \in \text{MSep}$ *— сепаратор, которому соответствует ровно одно владение. Никакие жилы не соответствуют такому сепаратору.*

***Определение* 1.20.** [19] *Стереосепаратор* $S \in \text{SSep}$ *— сепаратор, у которого как минимум два владения. Они разделяются на два вида: дисепаратор и полисепаратор* [19].

***Определение* 1.21.** [19] *Бездетный дисепаратор (или бисепаратор* $S \in \text{BSep}$*) — сепаратор, у которого ровно два владения и который не содержится ни в каком другом сепараторе. Множество жил бездетных дисепараторов образуют множество обязательных ребер* [15].

1.3. Схема алгоритмов синтеза подмножеств МГС

Для формирования ММГС строится промежуточное множество, называемое ModelKit, описанное в [15]. Для упрощения материала и реализации комплекса программ потребовалось другое определение, которое здесь будет приведено.

***Определение* 1.22.** *Через МК-пару будет обозначено множство следующего вида, а именно:* $\langle \text{NecessaryEdges}, \text{StereoHoldings} \rangle$ *— которое*

было построено по первичной структуре Workloads — *множеству нагрузок вершин.*

NecessaryEdges — *словарь, который возвращает обязательное ребро, т.е. ребро, которое встречается в каждом МГС, по соответствующему бисепаратору:*

$$\text{NecessaryEdges} = \{\langle S, E(\downarrow S)\rangle \mid S \in BSep\}$$

StereoHoldings *представляют собой словарь, которые возвращает стереовладения, или владения по заданному стереосепаратору.*

$$\text{StereoHoldings} = \{\langle S, \text{Components}(\twoheaddownarrow S)\rangle \mid S \in SSep\}$$

Пример первичной структуры и МК-пары, построенной для заданной структуры, изображены на Рис. 7–8.

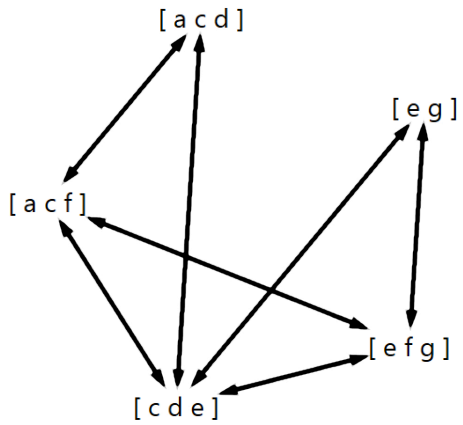
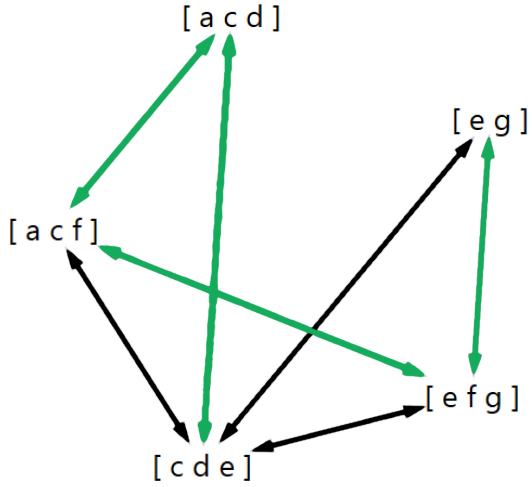


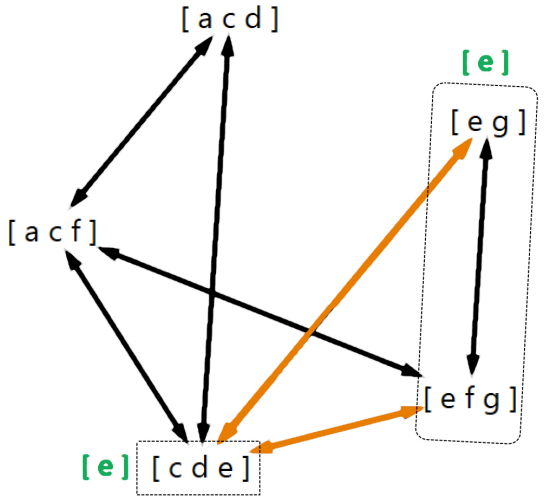
Рис. 7: Пример первичной структуры.

***Определение* 1.23.** *Инкрементальная МК-пара, или* IncModelKit — *это такая пара, которая содержит множество обязательных ребер, соответствующие стереосепараторы и стереовладения, которые появились с добавлением новой нагрузки вершины в первичную структуру.*

***Определение* 1.24.** *Декрементальная МК-пара, или* DecrModelKit — *четверка, в которой сохраняются все изменения по сепараторам, ко-*



(a) Обязательные ребра выделены зеленым цветом.



(b) Стереовладения только у сепаратора с нагрузкой [e]. Оранжевым отмечены ребра, которые будут использоваться для создания множества жил по вышеуказанному сепаратору.

Рис. 8: Пример МК-пары, сконструированной по множеству нагрузок вершин, изображенных на Рис. 7.

торые будут удаляться в связи с удалением нагрузки из первичной структуры.

***Определение* 1.25.** *Алгоритмы синтеза подмножества МГС по* ModelKit *будем называть алгоритмами сборки, или* Assembly*.*

***Определение* 1.26.** *Инкрементальная алгоритм сборки IncAssembly — это такой алгоритм сборки, который принимает две МК-пары, исходную и инкрементальную, а также набор нагрузок вместе с добавленной нагрузкой, предыдущие посчитанные жилы, и возвращает ММГС.*

***Определение* 1.27.** *Декрементальная алгоритм сборки DecrAssembly — это алгоритм сборки, аналогичный инкрементальному, только на вход подается не инкрементальная МК-пара, а декрементальная, и после этого высчитывается результирующее ММГС.*

1.4. Обозначения, используемые в листингах алгоритмов

При написании псевдокода алгоритмов использовались следующие обозначения, которые приведены ниже.

***Обозначение* 1.1.** *$\langle S_0, S_1, \ldots, S_k \rangle$ назовем структуру, хранящую k элементов.*

***Обозначение* 1.2.** *$|\ldots|$ обозначает мощность указанного множества.*

***Обозначение* 1.3. for each** *s* **in** *S — метод, перебирающий элементы s некоторого множества S.*

***Обозначение* 1.4.** *$S[i]$ — индексатор множества S, применяется либо как обращение к i-ому элементу множества (например, как в* Vertices[0]*), либо же как переход к значению из словаря по ключу (например, как в* StereoHoldings[s]*).*

***Обозначение* 1.5.** *$\cup=$ используется для добавления нового элемента в структуру. Если структура — множество, то элемент добавляется по общепринятым правилам добавления элемента в множество. В остальных случаях, это обычное добавление в структуру.*

***Обозначение* 1.6. SeparatorSet** *— набор алгоритмов генерации структуры, хранящей непустые сепараторы* Separators*, который может содержать словарь пересечений элементов первичной структуры* SeparatorTable*, словарь основных вершин* MainVertices *по сепараторам, а также* SNNEs *— множество ребер, входящих в сильное сужение в обратном графе* [15].

***Обозначение* 1.7. NarrowingSet** *— набор алгоритмов, реализующий различные сужения: сужение вершин $V_{S\subseteq}$ и сильное сужение $G \downarrow S$ графа G по заданному сепаратору S.*

***Обозначение* 1.8. ComponentSet** *— набор алгоритмов, который может сохранить в структуру компоненты связности данного графа и компоненты связности графа, обратного данному.*

***Обозначение* 1.9. SonSet** — *алгоритм, который по каждому сепаратору сохраняет множество соответствующих сыновей* [27].

***Обозначение* 1.10. SortSeparatorSet** — *алгоритм сортировки множества сепараторов либо в порядке возрастания мощностей соответствующих сепараторов, либо в порядке убывания.*

***Обозначение* 1.11. HoldingsBy4thStructure** — *алгоритм, который возвращает множество владений по четвертичной структуре* [15].

***Обозначение* 1.12. IsChildless** — *алгоритм, который проверяет, есть ли сын у данного сепаратора в третичной структуре.*

***Обозначение* 1.13. SinewSet** — *это любой алгоритм генерации множества жил* [15]. *Всего таких алгоритмов разработано* 3*: алгоритм генерации всевозможных жил на основе кода Прюфера* [26, 33]*, алгоритм случайной генерации жилы и алгоритм генерации звездчатой жилы.*

***Обозначение* 1.14.** *Объединяющей алгебраической сверткой набора множеств* $S = \{S_i\}_{i=1,\dots,n}$*, где* $\forall i\ S_i = \{S_i^j\}_{j=1,\dots,n_i}$ *называется*

$$\mathbf{UAF}(S) = \bigotimes_{i=1}^{n} S_i, \text{ где } A \otimes B = \bigotimes_{i=1,j=1}^{m,n} A_i \cup B_j$$

***Обозначение* 1.15. SaveIfContainsInSeparatorTable** — *алгоритм, который выбирает сепараторы из первого множества, содержащиеся в таблице сепараторов.*

Выводы по главе

В этой главе были рассмотрены базовые объекты теории АБС, введены новые определения и обозначения, которые использовались во всей работе.

2. Описание ранее разработанных алгоритмов

Введение

В этой главе будут представлены псевдокоды двухфазных алгоритмов генерации множества МГС: сперва происходит генерация МК-пары по первичной структуре, или множеству нагрузок, а затем по этой паре будет генерироваться ММГС с помощью алгоритма сборки. Корректность и сложность работы алгоритмов описаны в [16, 19].

Так как разрабатывающаяся библиотека рассматривается в контексте языка программирования C#, общий вид алгоритмов слегка изменится, однако суть их останется той же, что и в предыдущих работах.

2.1. МК-алгоритмы

2.1.1. Алгоритм МК1

В листинге 1 приведен псевдокод алгоритма генерации множества МГС МК1.

Algorithm 1 [15] Алгоритм генерации множества МГС МК1

Input: Workloads
Ensure: Workloads ∈ **PrimaryStructures**
Output: ⟨NecessaryEdges, StereoHoldings⟩

```
1:  ⟨Separators, SeparatorTable⟩ = SeparatorSet(Workloads, SO)
2:  for each s in Separators
3:      Vertices = NarrowingSet(s, Workloads)
4:      if |Vertices| = 2
5:          NecessaryEdges[s] = ⟨Vertices[0], Vertices[1]⟩
6:      else
7:          Holdings = ComponentSet(
8:              NarrowingSet(Vertices, s, SeparatorTable), Connected)
9:          if |Holdings| > 1
10:             StereoHoldings[s] = Holdings
```

Идея алгоритма МК1 в следующем: сначала строится множество непустых сепараторов (строка 4), затем для каждого сепаратора перебором всех нагрузок строится множество вершин, входящих в соот-

ветствующее сужение (строка 3). Для построенного множества вершин определяется, является ли сепаратор бисепаратором, то есть число вершин равно двум, или он является стереосепаратором, то есть не бисепаратор и владений больше одного. В первом случае (строки 4–5) ребро добавляется к множеству обязательных ребер, во втором случае владения строятся как компоненты связности сильного сужения (строки 7–10).

2.1.2. Алгоритм MK2

В листинге 2 приведен псевдокод алгоритма генерации множества МГС MK2.

Algorithm 2 [15] Алгоритм генерации множества МГС MK2

Input: Workloads
Ensure: Workloads ∈ **PrimaryStructures**
Output: ⟨NecessaryEdges, StereoHoldings⟩

```
 1: ⟨Separators, SeparatorTable⟩ = SeparatorSet(Workloads, SO)
 2: Sons = SonSet(Separators)
 3: SortedSeparators = SortSeparatorSet(Separators, AscendingLength)
 4: for each s in SortedSeparators
 5:     Vertices = NarrowingSet(s, Workloads)
 6:     if |Vertices| = 2
 7:         NecessaryEdges[s] = ⟨Vertices[0], Vertices[1]⟩
 8:     else
 9:         Holdings = HoldingsBy4thStructure(s, Sons, Vertices)
10:         if |Holdings| > 1
11:             StereoHoldings[s] = Holdings
```

Идея алгоритма MK2 совершенствует алгоритм построения множества МГС на основе клик-владений: сначала строится множество непустых сепараторов (строка 1), а затем над построенным множеством строится третичная структура (родительский граф) (строка 2). После чего, для каждого сепаратора строится соответствующее сужение (строка 6). Как и в алгоритме MK1, бисепараторы отсеиваются по числу вершин сужения (строки 7–8). Здесь владения строятся за счет поиска компонент связности в четвертичной структуре (строка 10). Обработка построенных владений не отличается от обработки в MK1 (строки 11–12).

2.1.3. Алгоритм MK2+

В листинге 3 приведен псевдокод алгоритма генерации множества МГС MK2+.

Algorithm 3 [15] Алгоритм генерации множества МГС MK2+

Input: Workloads
Ensure: Workloads ∈ **PrimaryStructures**
Output: ⟨NecessaryEdges, StereoHoldings⟩

```
 1: ⟨Separators, SeparatorTable⟩ = SeparatorSet(Workloads, SO)
 2: Sons = SonSet(Separators)
 3: SortedSeparators = SortSeparatorSet(Separators, DescendingLength)
 4: for each s in SortedSeparators
 5:     if |Vertices[s]| = 0
 6:         Vertices[s] = NarrowingSet(s, Workloads)
 7:     if |Vertices[s]| = 2
 8:         NecessaryEdges[s] = ⟨Vertices[s][0], Vertices[s][1]⟩
 9:     else
10:         for each c in Sons[s]
11:             if |Vertices[c]| = 0
12:                 Vertices[c] = NarrowingSet(c, Vertices[s])
13:         Holdings = HoldingsBy4thStructure(s, Sons, Vertices)
14:         if |Holdings| > 1
15:             StereoHoldings[s] = Holdings
```

Преимущества алгоритма MK2+ над MK2 в том, что нам нужно строить вершины за счет не всех нагрузок, а только нагрузок родителя, если он существует (строки 10–12)

2.1.4. Алгоритм MK3

В листинге 4 приведен псевдокод алгоритма генерации множества МГС MK3.

Идея алгоритма MK3 базируется на алгоритме построения множества МГС на основе самоуправляемых клик-собственников. Вместо последовательного способа построения множества непустых сепараторов и множеств вершин сужения, строится одновременно множество сепараторов и множество их основных вершин (строка 1). Затем каждый построенный сепаратор проверяется: если это бисепаратор (строки 4–6), то он добавляется ко множеству обязательных ребер, иначе строятся

Algorithm 4 [15] Алгоритм генерации множества МГС МК3

Input: Workloads
Ensure: Workloads ∈ **PrimaryStructures**
Output: ⟨NecessaryEdges, StereoHoldings⟩

```
1: ⟨Separators, MainVertices⟩ = SeparatorSet(Workloads, SaMV)
2: Sons = SonSet(Separators)
3: for each s in SortedSeparators
4:     if |MainVertices[s]| = 2
5:         if IsChildless(Sons[s], s)
6:             NecessaryEdges[s] = ⟨MainVertices[s][0], MainVertices[s][1]⟩
7:     else
8:         Holdings = ComponentSet
9:             (NarrowingSet(MainVertices[s], s, SeparatorTable), Connected)
10:        if |Holdings| > 1
11:            StereoHoldings[s] = Holdings
```

его владения за счет поиска компонент в сильном сужении на соответствующий сепаратор (строки 8–9) и далее обрабатывается как и ранее (строки 10–11).

2.1.5. Алгоритм МК3+

В листинге 5 приведен псевдокод алгоритма генерации множества МГС МК3+.

Algorithm 5 [15] Алгоритм генерации множества МГС МК3+

Input: Workloads
Ensure: Workloads ∈ **PrimaryStructures**
Output: ⟨NecessaryEdges, StereoHoldings⟩

```
1: ⟨ Separators, MainVertices, SNNEs ⟩ =
2:       SeparatorSet(Workloads, SaMVaSNNE)
3: Sons = SonSet(Separators)
4: for each s in SortedSeparators
5:     if |MainVertices[s]| = 2
6:         if IsChildless(Sons[s], s)
7:             NecessaryEdges[s] = ⟨MainVertices[s][0], MainVertices[s][1]⟩
8:     else
9:         Holdings = ComponentSet(MainVertices[s], s, SNNEs, Reversed)
10:        if |Holdings| > 1
11:            StereoHoldings[s] = Holdings
```

Алгоритм МК3+ отличается от МК3 тем, что вместе с сепараторами и множеством основных вершин строится граф, обратный сильному

сужению (строки 1–2). Затем владения строятся на основе компонент связности в обратном графе (строка 9).

2.1.6. Алгоритм MK4

В листинге 6 приведен псевдокод алгоритма генерации множества МГС MK4.

Algorithm 6 [15] Алгоритм генерации множества МГС MK4

Input: Workloads
Ensure: Workloads ∈ **PrimaryStructures**
Output: ⟨NecessaryEdges, StereoHoldings⟩

```
1: ⟨ Separators, MainVertices, SeparatorTable ⟩ =
2:       SeparatorSet(Workloads, SaMV)
3: Sons = SonSet(Separators)
4: SortedSeparators = SortSeparatorSet(Separators)
5: for each s in SortedSeparators
6:       Vertices[s] = MainVertices[s]
7:       if |Vertices[s]| = 2
8:             if Sons[s] = ∅
9:                   NecessaryEdges[s] = ⟨Vertices[0], Vertices[1]⟩
10:            else
11:                  for each c in Sons[s]
12:                        Vertices[s] ∪ = Vertices[c]
13:      else
14:            Holdings = HoldingsBy4thStructure(s, Sons, Vertices)
15:            if |Holdings| > 1
16:                  StereoHoldings[s] = Holdings
17:            else
18:                  for each c in Sons[s]
19:                        Vertices[s] ∪ = Vertices[c]
```

Идея MK4 состоит в следующем: сначала строится множество непустых сепараторов одновременно с множествами основных вершин таких сепараторов (строки 1–2), затем строится третичная структура (строка 3). Сепараторы перебираются по возрастанию (строка 4), если у сепаратора s две основных вершины и нет детей, то он является бисепаратором и обрабатывается соответствующим образом (строки 7–9). Если же у него есть дети, то он является моносепаратором типа 1 [15], и тогда множество его вершин строится за счет объединения вершин его сыновей с его основными вершинами (строки 11–12). В том же случае, если у сепаратора оказывается более двух основных вершин, то множество его

владений синтезируется за счет построения полусиблингового графа и поиска компонент связности в нем. Если владений больше одного, то сепаратор обрабатывается как стереосепаратор (строки 15–16). В том же случае, если владение одно, то такой сепаратор является моносепаратором, но не моносепаратором первого типа, и множество его вершин строится тем же способом, что и для моносепаратора типа 1 (строки 17–19). Такая обработка моносепараторов обоих типов нужна для того, чтобы при построении владений родителей моносепараторов учитывались все вершины моносепаратора, а не только те, которые являются его основными (согласно утверждению 3.32, множество основных вершин совпадает с множеством вершин, входящих в сужение, только для бисепараторов и стереосепараторов, но не для моносепараторов) [16].

2.2. Алгоритм сборки

В листинге 7 приведен псевдокод алгоритма генерации множества МГС МК4.

Algorithm 7 [15] Алгоритм сборки

Input: Workloads, ⟨NecessaryEdges, StereoHoldings⟩, T
Ensure: StereoSeparators ∪ NecessaryEdges ≠ ∅
Output: SMJG

```
1: for each ⟨s, h⟩ in StereoHoldings
2:     Sinews[s] = SinewSet(h, T)
3: EdgeSets = UAF(Sinews)
4: for each EdgeSet in EdgeSets
5:     SMJG ∪ = ⟨Workloads, EdgeSet ∪ NecessaryEdges⟩
6: if EdgeSets = ∅
7:     SMJG = ⟨Workloads, NecessaryEdges⟩
```

В данном алгоритм для каждого стереовладения строится соответствующие жилы (строки 1–2). Затем, они собираются, чтобы образовать всевозможные варианты ребер (строка 3), после чего полученные ребра объединяются с обязательными (строки 4–5). Если таких ребер нет, то мы строим базу (строки 6–7).

Выводы по главе

В этой главе были рассмотрены алгоритмы синтеза ММГС, разработанные в предыдущих работах, которые послужили основой для появления алгоритмов, описанных в следующей главе.

3. Описание инкрементальных и декрементальных алгоритмов

Введение

Так как набор фрагментов знаний может динамически изменяться, то потребовалось рассмотреть случай генерации множества минимальных графов смежности при внесении или при исключении нагрузки вершины. Как следствие, необходимо разработать инкрементальные и декрементальные алгоритмы [27] и доказать их корректность, что и представлено в данной главе. В качестве основы были использованы алгоритмы, описанные в предыдущей главе.

3.0.1. Добавление и удаление вершины

Пусть заданы набор нагрузок вершин, МК-пара, построенная по этому набору, и множество жил — по стереосепараторам МК-пары. Рассмотрим случай внесения или удаления вершины при имеющимся построенном множестве.

Обозначим вершину, которая будет добавляться или удаляться, как v, и обрабатываемый сепаратор, который в данный момент рассматривается, как s, тогда есть два варианта:

1. $W_s \not\subset v$
2. $W_s \subset v$

В первом случае вершина v никак не повлияла на существующий сепаратор вне зависимости от того, стереосепаратор он или бисепаратор. Поэтому пересчитывать ничего не нужно для этого случая.

Во втором случае тип сепаратора может поменяться. Допустим, у нас был бисепаратор $s \in \mathrm{BSep}$, и он порождал обязательное ребро e. Тогда необходимо пересчитать сужение на сепаратор s и определить тип s в новом наборе вершин, после чего соответственным образом обработать полученный результат. Если же у нас был стереосепаратор

$s \in SSep$, то число владений могло как возрасти (например, появилось ребро, нагрузка которого равна s, и один конец которого равен v, или, к примеру, удалилось ребро, которое связывало несколько владений), так и уменьшиться (допустим, появилось ребро, которое связывает разные владения, или предположим, удалилось ребро, нагрузка которого была равно s и один конец которого был равен v). Здесь также придется пересчитать сужение на сепаратор s в новом наборе вершин и посчитать владения, если мощность сильного сужения по s больше 2.

После построения инкрементальной или декрементальной МК-пары требуется посчитать жилы всех стереосепараторов, т.е. перестроить жилы тех сепараторов, владения которых изменились при изменении первичной структуры, сохранить предыдущие результаты тех стереосепараторов, которые не изменились при внесенных данных и удалить все те жилы, стереосепараторы которых исчезли после добавления изменений в нагрузки вершин.

3.1. Инкрементальные алгоритмы

3.1.1. Инкрементальный алгоритм построения множества непустых сепараторов

В листинге 8 приведен инкрементальный алгоритм построения множества непустых сепараторов.

Algorithm 8 Алгоритм построения множества непустых сепараторов IncSeparatorSet

Input: ⟨Separators, SeparatorTable⟩, Workloads, w'
Ensure: Workloads $\cup\, w' \in$ **PrimaryStructures**
Output: ⟨Separators′, SeparatorTable′⟩

```
1: SeparatorTable′ = SeparatorTable
2: for each w in Workloads
3:     SeparatorTable′[w, w′] = w ∩ w′
4:     if w ∩ w′ ≠ ∅ then
5:         Separators′ ∪ = SeparatorTable′[w, w′]
```

Идея алгоритма генерации непустых сепараторов в следующем: получив на вход набор сепараторов ⟨Separators, SeparatorTable⟩, который

был сформирован по указанным нагрузкам вершин Workloads, а также новую вершину w', алгоритм перебирает все нагрузки Workloads. Пересечение элементов $w \in Workloads$ и w' записывается в таблицу SeparatorTable$'$, и если пересечение непустое, то оно записывается в множество сепараторов Separators$'$. Таким образом, происходит сбор всех сепараторов, которые получаются при добавлении нагрузки w'.

3.1.2. Инкрементальный алгоритм генерации инкрементальной МК-пары

В листинге 9 приведен псевдокод инкрементального алгоритма MK1Inc генерации инкрементальной МК-пары.

Algorithm 9 Инкрементальный алгоритм MK1Inc генерации инкрементальной МК-пары

Input: ⟨Separators, SeparatorTable⟩, ⟨NecessaryEdges, StereoHoldings⟩, Workloads, w'
Ensure: Workloads $\cup$ $w' \in$ **PrimaryStructures**
Output: IncModelKit

```
1:  ⟨Separators′, SeparatorTable′⟩ =
2:      IncSeparatorSet(⟨Separators, SeparatorTable⟩, Workloads)
3:  for each s in Separators′
4:      Vertices = NarrowingSet(s, Workloads ∪ {w′})
5:      if |Vertices| = 2
6:          NecessaryEdges′[s] = ⟨Vertices[0], Vertices[1]⟩
7:      else
8:          Holdings = ComponentSet(NarrowingSet(Vertices, s, SeparatorTable′))
9:          if |Holdings| > 1 then
10:             StereoHoldings′[s] = Holdings
```

Суть алгоритма заключается в следующем: мы выделяем все сепараторы, которые будут задействованы после добавления нагрузки вершины w' (строки 1–2). Далее перебираются эти сепараторы. Строится сужение вершин по заданному сепаратору уже в новом наборе нагрузок вершин (строка 4). Если это обязательное ребро, то оно соответствующим образом обрабатывается (строки 5–6), иначе строятся владения сепаратора (строка 8). Если это стереосепаратор, то сохраняются стереовладения (строки 9–10).

3.2. Декрементальные алгоритмы

3.2.1. Декрементальный алгоритм построения множества непустых сепараторов

В листинге 10 приведен псевдокод декрементального алгоритма построения множества непустых сепараторов.

Algorithm 10 Алгоритм построения множества непустых сепараторов DecrSeparatorSet

Input: ⟨Separators, SeparatorTable⟩, Workloads, w'
Ensure: Workloads $\cup$ $w' \in$ **PrimaryStructures**
Output: ⟨Separators′, SeparatorTable′⟩

```
1: for each ⟨w1, w2⟩ in SeparatorTable
2:     if w1 ≠ w′ and w2 ≠ w′ then
3:         SeparatorTable′[w1, w2] = SeparatorTable[w1, w2]
4: Separators′ = SaveIfContainsInSeparatorTable
5:                 (SeparatorTable[w′], SeparatorTable′)
```

Идея алгоритма генерации непустых сепараторов в следующем: получив на вход набор сепараторов ⟨Separators, SeparatorTable⟩, который был сформирован по указанным нагрузкам вершин Workloads, а также исключаемую вершину w', алгоритм удаляет из таблицы все сепараторы, которые были построены с данной вершиной (строки 1–3). После чего, сохраняются все сепараторы, которые останутся после исключения этой нагрузки и которые могли бы повлиять на изменения в МК-паре, т.е. на формирование обязательных ребер или стереосепараторов (строки 4–5).

3.2.2. Декрементальный алгоритм генерации декрементальной МК-пары

В листинге 11 приведен псевдокод декрементального алгоритма MK1Decr генерации декрементальной МК-пары.

Суть алгоритма заключается в следующем: выделяются все сепараторы, которые будут задействованы после удаления нагрузки вершины w' (строки 1–2), и рассматриваются все сепараторы, которые не образовали до этого обязательные ребра или стереовладения (от английского

Algorithm 11 Декрементальный алгоритм MK1Decr генерации декрементальной MK-пары

Input: ⟨Separators, SeparatorTable⟩, ⟨NecessaryEdges, StereoHoldings⟩, Workloads, w'
Ensure: Workloads ∈ **PrimaryStructures**
Output: ModelKit

```
1: ⟨Separators′, SeparatorTable′⟩ =
2:     DecrSeparatorSet(⟨Separators, SeparatorTable⟩, Workloads)
3: NNENSS = SaveIfContainsInSeparatorTable
4:     (Separators \ (NecessaryEdges ∪ StereoHoldings), SeparatorTable′)
5: for each s in Separators′ ∪ NNENSS
6:     Vertices = NarrowedVertices(s, Workloads \{w′})
7:     if |Vertices| = 2
8:         NecessaryEdges′[s] = ⟨Vertices[0], Vertices[1]⟩
9:     else
10:        Holdings = Components(StrongNarrowing(Vertices, s, SeparatorTable′))
11:        if |Holdings| > 1 then
12:            StereoHoldings′[s] = Holdings
```

Neither NecessaryEdges Nor StereoSeparators). Далее перебираются эти сепараторы. Строится сужение вершин по заданному сепаратору уже в новом наборе нагрузок вершин с учетом удаленной вершины (строка 6). Если это обязательное ребро, то оно соответствующим образом обрабатывается (строки 7–8), иначе строятся владения сепаратора (строка 10). Если это стереосепаратор, то сохраняются их владения (строки 11–12).

3.2.3. Инкрементальный и декрементальный алгоритм сборки

В листинге 12 приведен псевдокод инкрементального и декрементального алгоритма сборки IaDAssembly.

Алгоритм состоит из двух частей. В первой части происходит дополнение построенного множества обязательных ребер и множества стереовладений. Сначала проверяются все обязательные ребра, которые были построены до добавления нагрузки в первичную структуру (строка 1). Если у сепаратора, который образует данное ребро, не изменился тип, следует внести этот сепаратор в множество бисепараторов (строки 2–3). После этого проверяются все стереовладения (строка 4). Если стереовладения не расширились (строка 5) и новая вершина никак не повлияла на данный сепаратор (строка 6), сохраняем стереовладения и жилы (строки 7–8), а если расширились, надо пересчитать жилы (стро-

Algorithm 12 Инкрементальный и декрементальный алгоритм сборки IaDAssembly

Input: ⟨NecessaryEdges, StereoHoldings⟩, ⟨NecessaryEdges′, StereoHoldings′⟩, Workloads, w', SinewSet, T
Output: SMJG

```
 1: for each ⟨s, e⟩ in NecessaryEdges
 2:     if s ⊄ w′
 3:         NecessaryEdges′ ∪ = ⟨s, e⟩
 4: for each ⟨s, h⟩ in StereoHoldings
 5:     if s ∉ StereoHoldings′
 6:         if s ⊄ w′
 7:             StereoHoldings′ ∪ = ⟨s, h⟩
 8:             SinewSet′[s] = SinewSet[s]
 9:     else
10:         SinewSet′[s] = SinewSet(h, T)
11: for each ⟨s, h⟩ in StereoHoldings′ \ StereoHoldings
12:     SinewSet′[s] = SinewSet(h, T)
13: EdgeSets = UAF(SinewSet′)
14: SMJG = ∅
15: for each edgeSet in EdgeSets
16:     SMJG ∪ = ⟨Workloads ∪ w′, NecessaryEdges′ ∪ Edges⟩
17: if EdgeSets = ∅
18:     SMJG = ⟨Workloads ∪ w′, NecessaryEdges′⟩
```

ки 9–10). Далее, если не были рассмотрены какие-то уже построенные сепараторы, то считаем жилы для них (строки 11-12). Затем действия повторяются как и в алгоритме сборки, приведенном в листинге 7.

3.2.4. Обоснование корректности инкрементального алгоритма

***Теорема* 3.1.** *Множество* SMJG*, полученное применением инкрементального МК-алгоритма и инкрементальной сборки при внесении новой информации в первичную структуру, есть множество МГС.*

Доказательство. По определению, у стереосепараторов не менее двух жил, когда как у бисепараторов — ровно одна. Все ребра, не являющиеся обязательными, обязательно содержатся в жиле какого-нибудь стереосепаратора.

Все найденные бисепараторы, которые породили новые обязательные ребра при добавлении новой нагрузки вершины, были учтены в

инкрементальном МК-алгоритме, а те бисепараторы, которые не изменились при внесении нового ФЗ, были добавлены в алгоритме сборки в Листинге 12 (строка 3).

Если рассматривать стереосепараторы, то те, что образовались при добавлении новой нагрузки вершины, были учтены в инкрементальном МК-алгоритме, а стереосепараторы, владения которых остались прежними, были добавлены в алгоритме сборки из Листинга 12 (строка 7).

Затем нужно построить все жилы стереосепараторов. Если стереосепаратор не изменился, то нужно просто добавить имеющиеся, уже посчитанные жилы (строка 8). Если владения стереосепаратора изменились, нам придется пересчитать жилы (строки 10, 11–12).

На шаге 13 строится множество возможных наборов ребер каждого стереосепаратора. Затем в цикле 15–16 такие наборы объединяются с обязательными ребрами, то есть представляют из себя кортеж жил, выбранных по одному для каждого стереосепаратора. Согласно теореме о множестве минимальных графов смежности [15], таким образом будут перебраны все минимальные графы смежности. □

3.2.5. Обоснование корректности декрементального алгоритма

***Теорема* 3.2.** *Множество* SMJG*, полученное применением декрементального МК-алгоритма и декрементальной сборки при удалении информации из первичной структуры, есть множество МГС.*

Доказательство. Аналогично, рассматриваем случай удаления вершины из набора ФЗ.

Все новообразовавшиеся обязательные ребра при удалении нагрузки вершины были учтены в декрементальном МК-алгоритме, а все обязательные ребра, на которые никак не повлияло извлечение ФЗ, были добавлены в алгоритме сборки из Листинга 12 (строка 3).

Подобным образом разбираются стереосепараторы: те стереосепараторы, что образовались при удалении нагрузки, были учтены в декрементальном МК-алгоритме, а неизменившиеся были добавлены уже в декрементальном алгоритме сборки из Листинга 12 (строка 7).

Жилы сепараторов строятся по тому же образу и подобию: сохраняем ранее полученные результаты, если стереосепаратор не изменился (строка 8), иначе — считаем жилы на основе полученной информации (строки 10, 11–12).

На шаге 13 строится множество пучков, которое затем объединяется с множеством обязательных ребер. Полученное множество впоследствие будет содержать ребра МГС. Согласно теореме о множестве МГС [15], таким образом будут перебраны все МГС. □

Выводы по главе

В этой главе была решена одна из поставленных задач, а именно были рассмотрены разработанные алгоритмы генерации ММГС и представлены теоремы, доказательства которых подтверждают корректность работы этих алгоритмов.

4. Статистическая оценка сложности алгоритмов

Введение

Чтобы оценить эффективность разработанных алгоритмов, необходимо сопоставить их с существующими алгоритмами, для чего и был проведен сравнительный анализ сложностей. Как результат, в данной главе будут приведены результаты вычислительных экспериментов, представленных в виде графиков.

4.1. Схема проведения экспериментов

Для успешного осуществления экспериментов над вышеуказанными алгоритмами использовалось три множества: набор алфавитов, список распределений мощностей нагрузок и множество количества вершин. Также имеется GenerateNodes — алгоритм генерации нагрузок вершин по указанному алфавиту, распределению мощностей нагрузок и количеству вершин. Параметром сравнения выступает время работы двух алгоритмов, которое требовалось для синтеза ММГС по полученной первичной структуре. С детальным описанием методики статистической оценки сложности и компаративного анализа можно познакомиться в [5].

4.2. Результаты

Эксперименты проводились над первичной структурой мощности от 4 до 12 вершин с шагом 1[1]. Результаты экспериментов по сравнению скоростей работ инкрементального и прямого алгоритмов синтеза ММГС представлены на Рис. 9–10, а работы декрементального и прямого алгоритмов — на Рис. 11–12.

Графики представляют выходные данные алгоритма Experiment [4]. В качестве статистических характеристик были выбраны следующие

[1]Выбранный диапазон связан с мощностью получающегося ММГС [3].

величины: MIN и MAX — минимальные и максимальные значения времен соответственно, D1 и D9 — первый и девятый децили соответственно [2], Q1 и Q3 — первый и третий квартили, а также Rg — среднее геометрическое. По горизонтальной оси откладывается мощность первичной структуры, а по вертикали — отношение работ инкрементального алгоритма к прямому для первых двух рисунков и отношение работ декрементального алгоритма к прямому для последних двух.

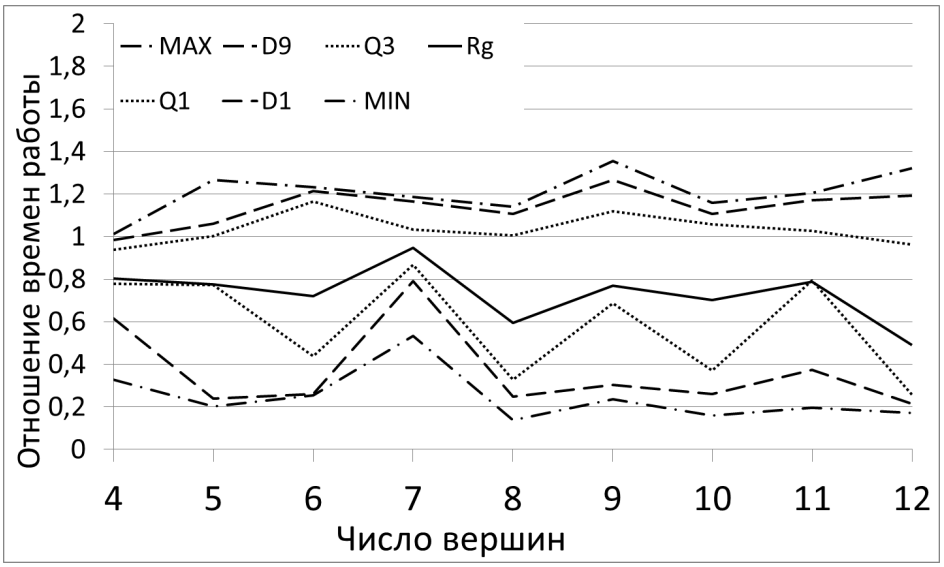


Рис. 9: Прямой и инкрементальный алгоритмы. Алфавит — 61 символ. 85% вершин — 2–4 порядков, 12% — 5–7 порядков, 3% — 8–10 порядков.

Анализ первых двух графиков, представленных на Рис. 9–10, позволяет заключить, что в среднем инкрементальный алгоритм превосходит по временным показателям прямой алгоритм (траектория среднего геометрического), однако существуют наборы, на которых инкрементальный алгоритм незначительно уступает прямому (траектория максимума).

Анализ последних двух графиков, изображенных на Рис. 11–12, подтверждает, что декрементальный алгоритм работает быстрее прямого алгоритма, о чем свидетельствует тот факт, что траектории всех статистических величин лежат под графиков $y = 1$.

Приведенные рисунки не отображают признаки монотонного убывания или возрастания величины Rg на всем диапазоне, однако в некото-

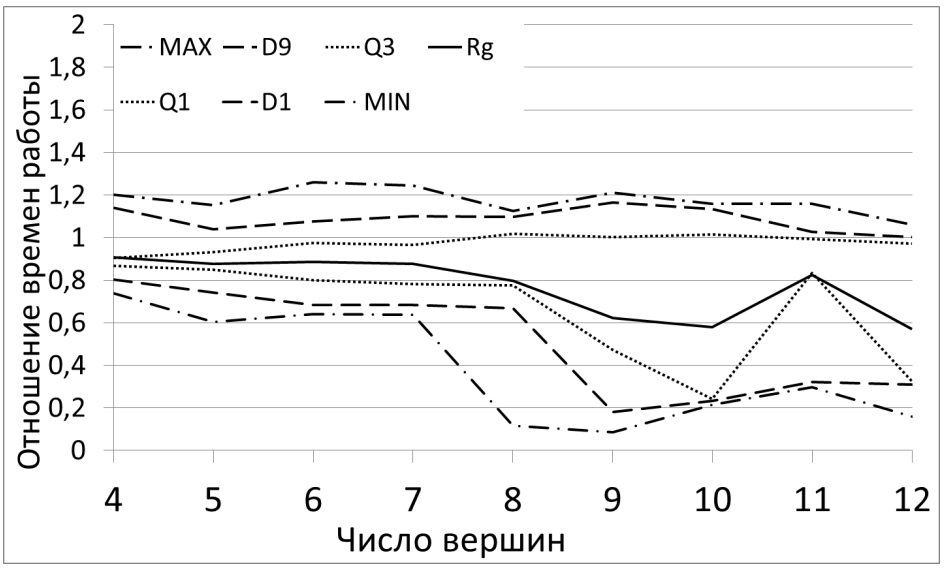


Рис. 10: Прямой и инкрементальный алгоритмы. Алфавит — 61 символ. 77% вершин — 2–4 порядков, 15% — 5–7 порядков, 5% — 8–10 порядков, 3% — 11–13 порядков.

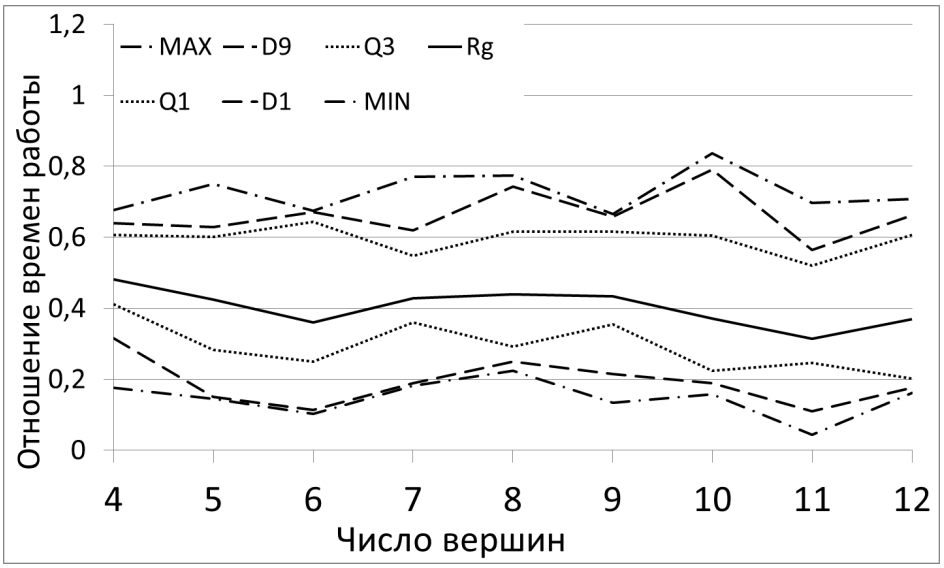


Рис. 11: Прямой и декрементальный алгоритмы. Алфавит — 61 символ. 85% вершин — 2–4 порядков, 12% — 5–7 порядков, 3% — 8–10 порядков.

рых случаях можно отметить некоторую тенденцию на поддиапазоне. Например, на Рис. 12 на поддиапазоне вершин 8–12 при возрастании количества вершин отношение времени работ монотонно уменьшается.

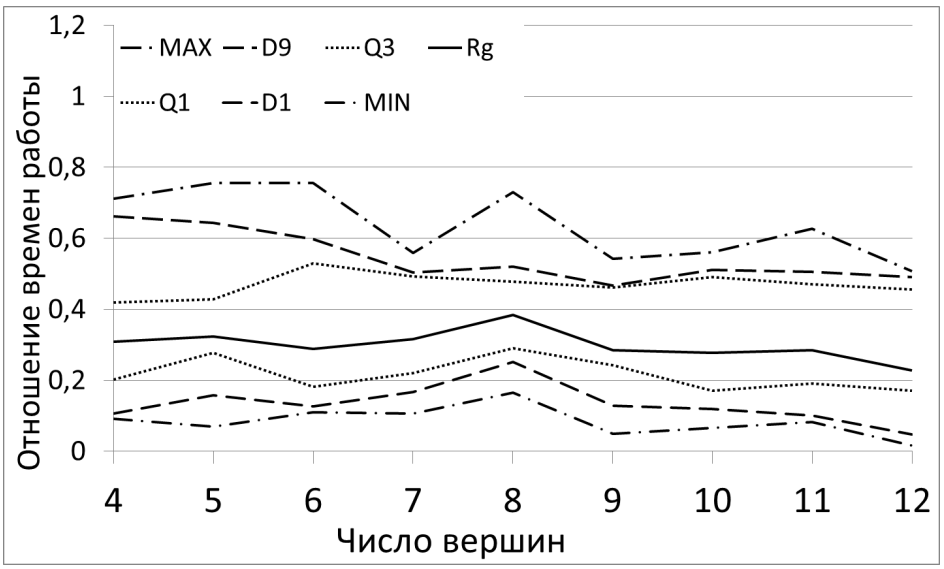


Рис. 12: Прямой и декрементальный алгоритмы. Алфавит — 61 символ. 77% вершин — 2–4 порядков, 15% — 5–7 порядков, 5% — 8–10 порядков, 3% — 11–13 порядков.

Заключение

В данной главе была представлена визуализация статистических тестов, показывающая, что на заданных диапазонах вершин разработанные алгоритмы опережают существующие.

5. Система анализа и синтеза

Введение

В данной главе приводится описание архитектура компоненты комплекса программ с детальным описанием всех содержащихся структур. После чего приведены примеры визуализаций данных структур и работы инкрементальных и декрементальных алгоритмов.

5.1. Архитектура

Данная выпускная квалификационная работа описывает систему, позволяющую пользователям генерировать и анализировать различные глобальные структуры АБС. Разработанный комплекс программ предназначен помочь студентам, которые изучают алгебраические байесовские сети, а также логико-вероятностный вывод: во-первых, для ускоренного восприятия материала при помощи визуалиции, а во-вторых, для дальнейших разработок в данной области.

Для написания библиотеки использовались язык программирования C#, интегрированная среда разработки Microsoft Visual Studio 2015. Целесообразность использования данных средств заключена в кроссплатформенности, расширяемости системы и высокой скорости работы.

Данная работа взяла основы предыдущих проектов [11, 12].

Часть комплекса программ, которой посвящана данная работа, представлена в виде следующих классов:

1. MKDataClasses
2. MKData

MKDataClasses описывает объекты теории алгебраических байесовских сетей, а именно:

1. Множество сепараторов, а также виды конструирования этого множества.
2. Множество компонент связности и их тип построения.

3. Множество сужений.

4. Код Прюфера.

5. Множество жил и перечислитель их реализаций.

6. МК-пара с ее различными вариациями.

Класс MKData содержит основную функциональность, ответственную за синтез множества минимальных графов смежности. Конструктор структуры MKData принимает различные представления первичной структуры. В данной структуре содержится следующий функционал:

1. Поле LoadSet, представляющее собой список, — первичная структура.

2. Поле SeparatorSet возвращает структуру типа SeparatorSet.

3. Поле StrongNarrowing возвращает словарь, где по каждому сепаратору возвращается соответствующее сильное сужение типа NarrowingSet.

4. Поле NarrowedVertices хранит словарь, где по каждому сепаратору изымается соответствующее сужение вершин типа NarrowingSet.

5. Поле Quadruple возвращает МК-пару типа Quadruple

6. Поле SinewsType хранит тип построенных жил.

7. Поле SinewSet представляет из себя словарь, который по каждому стереосепаратору получает двухмерный список ребер.

8. Поле ChangedData возвращает MKData — результат работы инкрементального и декрементального алгоритмов на данном шаге, если таковые имеются.

9. Поле ChangedLoad возвращает вершину, которая будет добавлена или удалена.

10. Поле SMJG возвращает само ММГС.

11. Методы внесения и удаления вершин AddLoad и RemoveLoad.

12. Методы GetMK*, которые возвращают MK-пару по соответствующему MK-алгоритму.

13. Методы GetMK*Inc и GetMK*Decr, которые возвращают инкрементальные и декрементальные MK-пары соответственно.

14. Методы Assembly, IncAssembly и DecrAssembly, которые собирают полученные обычные, инкрементальные и декрементальные MK-пары соответственно в ММГС.

5.2. Визуализация

Ниже приведены примеры графического пользовательского интерфейса, в частности, пример МГС до и после внесения нагрузки вершины $[\ a\ f\ ]$, после чего множество МГС было перестроено (Рис. 13–14).

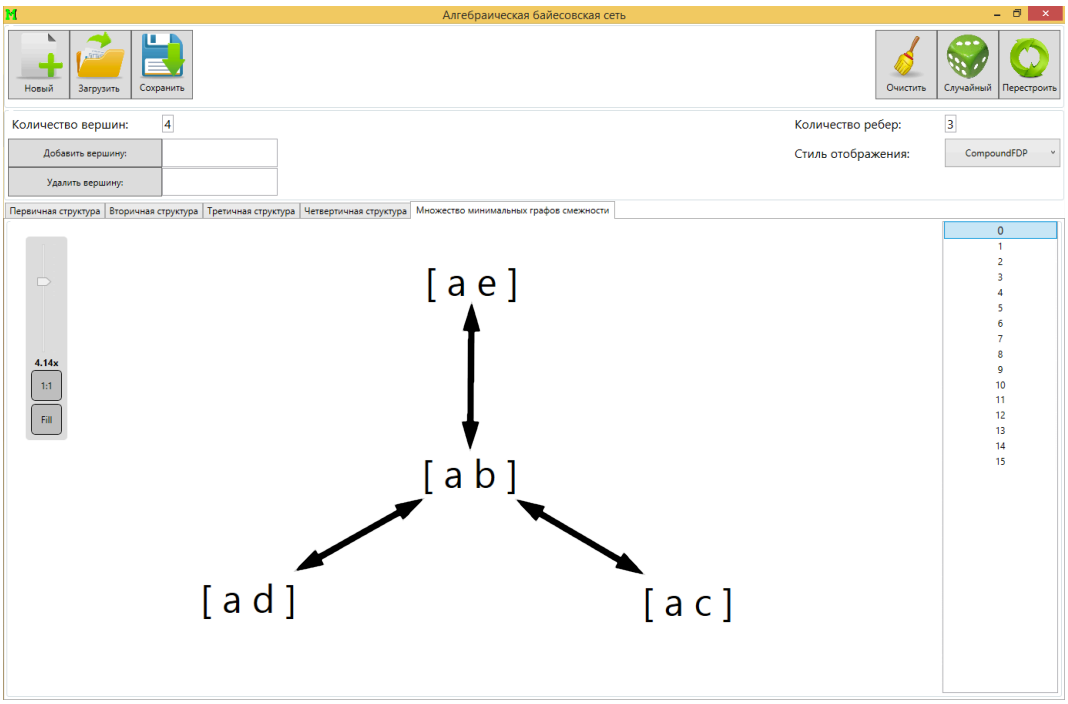


Рис. 13: Пример визуализации ММГС

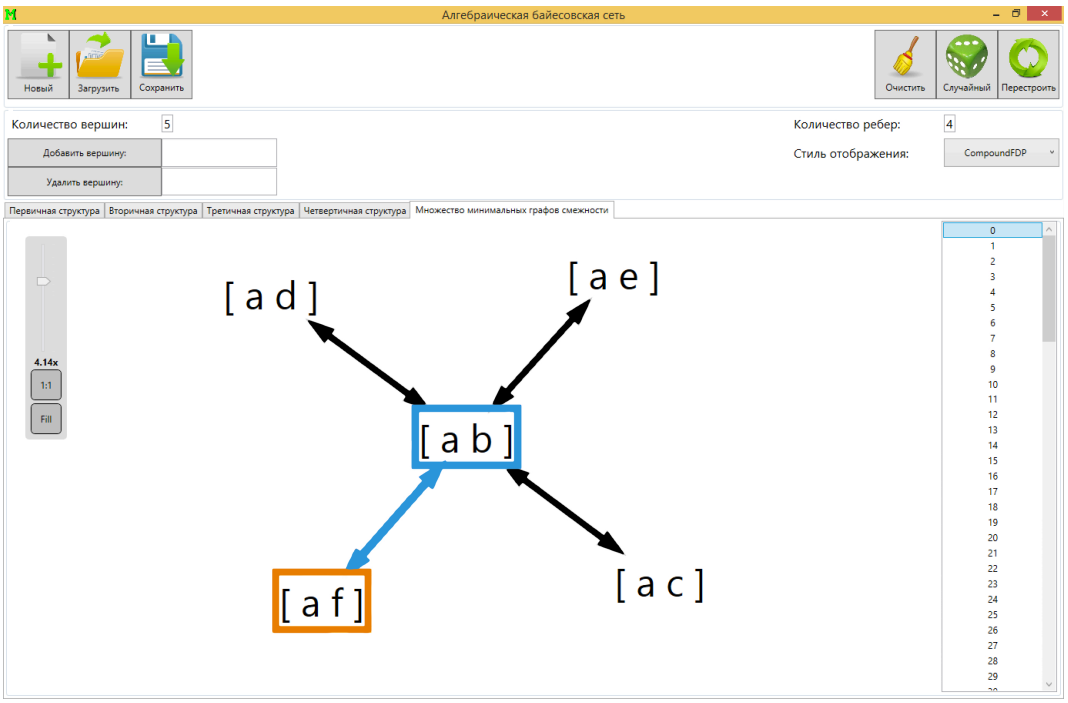


Рис. 14: Пример визуализации ММГС

На полученных рисунках можно увидеть результат добавления новой вершины в первичную структуру. Сначала у нас имелось 4 вершины. Количество графов в таком случае — n^{n-2}, где n — количество вершин [15, 26]. Данная формула верна в связи с единственным стереосепаратором и отсутствием обязательных ребер. В нашем случае мощность ММГС равна $4^{4-2} = 4^2 = 16$. После внесения вершины количество обязательных ребер и стереосепараторов не изменилось. Тогда данная формула остается верна, и мощность будет равна $5^{5-2} = 5^3 = 125$.

Выводы по главе

В данной главе была приведена архитектура компоненты комплекса программ и представлена визуализация работы разработанных алгоритмов.

Заключение

В настоящей работе были представлены базовые определения глобальных вторичных структур и ММГС, приведены существующие алгоритмы генерации необходимого множества, описана архитектура полученного программного комплекса с диаграммами написанных классов, а также приведены примеры работы вышеописанных алгоритмов.

В результате проделанной работы реализована библиотека на C#, состоящая из различных компонент и имеющая разнообразную функциональность. Функциональность данной программного комплекса покрывает большую часть операций, существующих на данных момент для синтеза ММГС. Данная генерация представляет научный интерес и даст отличную возможность при реализации глобального вывода.

Кроме того, вышеприведенные результаты позволят интегрировать воедино реализации алгоритмов представления и обработки знаний с неопределенностью на основе байесовских сетей доверия.

Перспективными направлениями дальнейших исследований являются направления, связанные с дальнейшей оптимизацией генерации множества минимальных графов смежности, а также поиска минимальных графов смежности, обладающих оптимальными численными характеристиками для проведения глобального логико-вероятностного вывода.

Список литературы

[1] Березин А. И. Множество минимальных графов смежности: примеры использования программного комплекса на языке C#. // Гибридные и синергетические интеллектуальные системы. 2016 [в печати].

[2] Березин А. И., Зотов М. А., Иванова А. В. Синтез множества минимальных графов смежности: статистическая оценка сложности декрементального алгоритма. // Нечеткие системы и мягкие вычисления. 2016 [в печати].

[3] Березин А. И., Романов А. В., Зотов М. А. Синтез вторичной структуры алгебраических байесовских сетей: оценка мощности множества минимальных графов смежности. // Нечеткие системы и мягкие вычисления. 2016 [в печати].

[4] Зотов М. А., Левенец Д. Г., Тулупьев А. Л., Золотин А. А. Синтез вторичной структуры алгебраических байесовских сетей: инкрементальный алгоритм и статистическая оценка его сложности. // Научно-технический вестник информационных технологий, механики и оптики. 2016. Т. 16. № 1. С. 122–132.

[5] Зотов, М. А., Тулупьев, А. Л. Синтез вторичной структуры алгебраических байесовских сетей: методика статистической оценки сложности и компаративный анализ прямого и жадного алгоритмов. // Компьютерные инструменты в образовании. 2015. № 1. С. 3–16.

[6] Зотов М. А., Тулупьев А. Л., Сироткин А. В. Статистические оценки сложности прямого и жадного алгоритмов синтеза вторичной структуры алгебраических байесовских сетей. // Нечеткие системы и мягкие вычисления. 2015. Т. 10. № 1. С. 75–91.

[7] Тулупьев А. Л. Алгебраические байесовские сети: реализация

логико-вероятностного вывода в комплексе Java-программ. // Труды СПИИРАН. 2009. Вып. 8. С. 191–232.

[8] Тулупьев А. Л., Николенко С. И., Сироткин А. В. Байесовские сети: логико-вероятностный подход. // СПб.: Наука. 2006. С. 607.

[9] Тулупьев А. Л., Сироткин А. В., Николенко С. И. Байесовские сети доверия. // СПб.: Изд-во СПбГУ. 2009. С. 400.

[10] Тулупьев А. Л., Столяров Д. М., Ментюков М. В. Представление локальной и глобальной структуры алгебраической байесовской сети в Java-приложениях. // Труды СПИИРАН. 2007. Вып. 5. С. 71–99. Доступна онлайн: http://mi.mathnet.ru/trspy301.

[11] Тулупьев А. Л., Фильченков А. А., Сироткин А. В. Система для построения и визуализации семейства минимальных вторичных структур алгебраической байесовской сети, соответствующих ее первичной структуре Algebraic Bayesian Networks Secondary Structures Generator, Version 1 for Java (AlgBNSSGj.v.01). // Свид. о гос. рег. эл. ресурса (ОФЭРНиО ИИО ГАН РАО) № 15769 от 20.05.2010. Бюлл. "Хроники объед. фонда эл. ресурсов "Наука и образование". 2010. № 5. С. 27.

[12] Тулупьев А. Л., Фильченков А. А., Сироткин А. В. Программа для синтеза семейства минимальных графов смежности Algebraic Bayesian Networks Secondary Structures Generator, Version 1 for Java (AlgBNSSGj.v.01) // Свид. о гос. рег. прогр. для ЭВМ. Рег. № 2010614269 (30.06.2010). РОСПАТЕНТ. Бюлл. "Прогр. для ЭВМ, БД, тол. инт. микросх.". 2010. № 3. С. 455–456.

[13] Фильченков А. А. Алгоритмы построения третичной структуры алгебраической байесовской сети. // Труды СПИИРАН. 2011. Вып. 17. С. 197–218.

[14] Фильченков А. А. Преобразование первичной структуры алгебраической байесовской сети к ациклической с сохранением вероятностной семантики. // Труды СПИИРАН. 2013. Вып. 30. С. 156–168.

[15] Фильченков А. А. Синтез графов смежности в машинном обучении глобальных структур алгебраических байесовских сетей. // СПбГУ. 2013.

[16] Фильченков А. А., Тулупьев А. Л. Структурный анализ систем минимальных графов смежности. // Труды СПИИРАН. 2009. Вып. 11. С. 104–129. Доступна онлайн: http://mi.mathnet.ru/trspy50.

[17] Фильченков А. А., Тулупьев А. Л. Алгоритм выявления ацикличности первичной структуры алгебраической байесовской сети по ее четвертичной структуре. // Труды СПИИРАН. 2011. Вып. 19. С. 128–145.

[18] Фильченков А. А., Тулупьев А. Л. Связность и ацикличность первичной структуры алгебраической байесовской сети. // Вестник СПбГУ. 2013. Серия 1: Математика, Механика, Астрономия. № 1. С. 110–118.

[19] Фильченков А. А., Фроленков К. В., Сироткин А. В., Тулупьев А. Л. Система алгоритмов синтеза пожмножеств минимальных графов смежности. // Труды СПИИРАН. 2013. Вып. 4. № 27.

[20] Фильченков А. А., Фроленков К. В., Тулупьев А. Л. Устранение циклов во вторичной структуре алгебраической байесовской сети на основе анализа ее четвертичной структуры. // Труды СПИИРАН. 2012. Вып. 21. С. 143–156.

[21] Фроленков К. В., Фильченков А. А., Тулупьев А. Л. Апостериорный вывод в третичной полиструктуре алгебраической байесовской сети. // Труды СПИИРАН. 2012. Вып. 23. С. 343–356.

[22] Dlamini W. M. A Bayesian belief network analysis of factors influencing wildfire occurrence in Swaziland. // Environmental modelling & software. 2010. Vol. 25. pp. 199–208.

[23] Farasat A., Nikolaev A., Srihari S. N., Blair R. H. Probabilistic

graphical models in modern social network analysis. // Plos Computational Biology. 2015. Vol. 5. no. 1.

[24] Farine D. R., Strandburg-Peshkin A. Estimating uncertainty and reliability of social network data using Bayesian inference. // Royal Society Open Science. 2015. Vol. 2. no. 9. pp. 150–367.

[25] Kappel D., Habenschuss S., Legenstein R., Maass W. Network Plasticity as Bayesian Inference. // Plos Computational Biology. 2015. Vol. 11. no. 11.

[26] Lennon Craig. On the Locality of the Prufer Code. // Electronic Journal of Combinatorics. 2009. Vol. 16. no. R10. pp. 197–218.

[27] Levenets D. G., Zotov M. A., Romanov A. V., Tulupyev A. L., Zolotin A. A., Filchenkov A. A. Decremental and Incremental Reshaping of Algebraic Bayesian Networks Global Structures. // Proceedings of IITI'16. May 16—-21, 2016, Sochi, Russia [in print].

[28] Mal'chevskaya E. A., Berezin A. I., Zolotin A. A., Tulupyev A. L. Algebraic Bayesian Networks: Local Probabilistic-Logic Inference Machine Architecture and Set of Minimal Joint Graphs. // Proceedings of IITI'16. May 16—-21, 2016, Sochi, Russia [in print].

[29] Nelson A. W., Malik A. S., Wendel J. C., Zipf M. E. Probabilistic Force Prediction in Cold Sheet Rolling by Bayesian Inference. // Journal of Manufacturing Science and Engineering-Transactions of the Asme. 2014. Vol. 136. no. 4.

[30] Razib R. H., Nikolaev A. Probabilistic Graphical Modeling method for inferring hydraulic conductivity maps from hydraulic head maps. // Journal of Hydrologic Engineering. 2016. Vol. 21. no. 2.

[31] Richards R. G., Sano M., Sahin O. Exploring climate change adaptive capacity of surf life saving in Australia using Bayesian belief networks. // Ocean & Coastal Management. 2016. Vol. 120. pp. 148–159.

[32] Yodo N., Wang P. Resilience Modeling and Quantification for Engineered Systems Using Bayesian Networks. // Journal of Mechanical Design. 2016. Vol. 138. no. 3.

[33] Seo Seunghyun, Shin Heesung A generalized enumeration of labeled trees and reverse Prufer algorithm. // Journal of Combinatorics Theory Series A. 2007. Vol. 114. no 7. pp. 1357–1361.